AD-A068 362 NAVAL RESEARCH LAB WASHINGTON D C F/G 9/2
EDIT: A FORTRAN PROGRAM MAINTENANCE SYSTEM FOR THE TEXAS INSTRU--ETC(U)
APR 79 P MORAWSKI

UNCLASSIFIED NRL-MR-3978 NL

| OF |
AD
AD 3?2

END
DATE FILMED
6-79
DDC

LEVEL II

# EDIT: A FORTRAN Program Maintenance System for the Texas Instruments ASC-7

PAUL MORAWSKI

*Software Systems and Support Branch*
*Research Computation Center*

AD A068362

DDC FILE COPY

DDC
RECEIVED
MAY 8 1979
C

April 16, 1979

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (14) NRL-MR-
NRL Memorandum Report 3978

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER (9)

4. TITLE (and Subtitle) (6)
EDIT: A FORTRAN PROGRAM MAINTENANCE SYSTEM FOR THE TEXAS INSTRUMENTS ASC-7

5. TYPE OF REPORT & PERIOD COVERED
Interim report on a continuing NRL problem

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s) (10)
Paul Morawski

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Research Laboratory
Washington, DC 20375

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
NRL General and Admin.
Function 71756-7215

11. CONTROLLING OFFICE NAME AND ADDRESS (11) 16

12. REPORT DATE
April 6, 1979

13. NUMBER OF PAGES
16

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 17p.

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This report supersedes an earlier report having the same title and author (David Taylor Naval Ship Research & Development Center Report CMLD-77-07, March, 1977).

DTNSRDC

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Program Maintenance
Advanced Scientific Computer (ASC)
Macro Programs
Job Specification Language

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

→ EDIT is a MACRO program for automatic maintenance of FORTRAN and RATFOR (RATional FORTRAN) programs on the Texas Instruments Advanced Scientific Computer. It has been used extensively to accelerate the creation of complex fluid dynamics programs for execution on the Advanced Scientific Computer (ASC).

EDIT allows the user to more efficiently create and modify programs using the TI-supplied Source Management System, SMS. Object libraries and load modules may also be automatically

(Continues)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



251950

20. Abstract (Continued)

updated. The program files (source, object, and load module) may be tape or disk resident and will always have a user-specified number of versions retained as backup. The EDIT system is described and sample usages are presented.

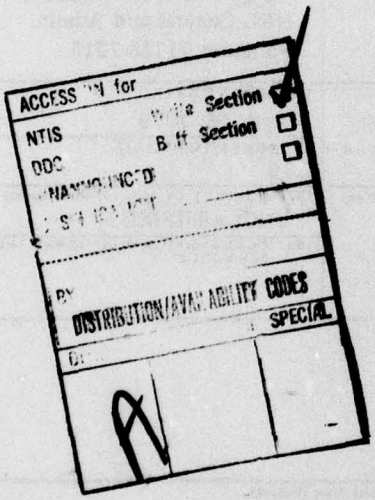

Table of Contents

# EDIT: A FORTRAN PROGRAM MAINTENANCE SYSTEM FOR THE TEXAS INSTRUMENTS ASC—7

## INTRODUCTION

The Texas Instruments Advanced Scientific Computer (ASC) is a large-scale digital computer which was installed at the Naval Research Laboratory in April 1976. Its vector arithmetic capability and massive central memory make it attractive for use in solving large numerical problems. While the ASC was being used to develop involved fluid mechanics programs, the need for some procedure to automatically update programs became apparent. This deficiency led to the development of a MACRO program, EDIT, which generates the control cards a user needs to update a program library. One EDIT call and a set of Texas Instruments Source Management System (SMS) directives allow a programmer to perform a program update which, if done manually using individual Job Specification Language (JSL) statements, would require approximately 80 control cards.

This documentation assumes that the reader has some familiarity with both the Texas Instruments Job Specification Language and their Source Management System. Users may consult the JSL Reference Manual and the Source Management System User's Guide for further clarification; both of these manuals are available through NRL Code 1722.

EDIT was modeled on a similar program used on the CDC 6000 computers at the David Taylor Naval Ship Research and Development Center (DTNSRDC). This program, also called EDIT, was developed by Mel Haas of DTNSRDC, Code 1843, in 1972.

**Note: Manuscript submitted February 15, 1979.**

## DESCRIPTION

The EDIT macro utilizes TI-supplied software (eg., SMS, CIFER, FTN, PDSQSH, etc.) to automate the maintenance of FORTRAN and RATFOR* routines. EDIT stores the source code, object code, and load module for a given program as ASC-cataloged files with a user-specified number of versions.

To use EDIT to maintain a program the user must first catalog a node (i.e., filename) in the ASC tree-structured file catalog system. The user may then use EDIT to perform a "creation run" (indicated by EDIT's OLDSRC and OLDOBJ parameters) to establish a program library. EDIT will add three sons to the user-supplied node: SRCLIB, OBJLIB, and LOADLIB, for the source code, object code, and load module, respectively. These nodes will have partial access control and full son-add control†. The maximum number of versions of the files is specified by the MXVR parameter on the initial macro call.

EDIT will catalog files containing the source code, object code, and load module, depending on the values of the EDIT parameters IEDIT and CAT. However, these files will not be cataloged if any of the software called by EDIT ends with a termination code which indicates a fatal error. Each time a new version of a file is cataloged, it is automatically flagged by the system as the version to be used by the next EDIT run. When the user-specified number of versions has been cataloged, the oldest version will be replaced by the new version. Thus, a programmer may update and maintain programs simply by supplying update information to EDIT. All file manipulation and library updating are managed by the EDIT macro.

---

* RATional FORTRAN preprocessor, see: Kernighan and Plauger, "Software Tools," Addison-Wesley (1976).

† These terms are used to define specific characteristics of a node. They control who may access the node and who may add sons beneath the node.

## INPUT TO EDIT

Input to EDIT is in the form of SMS directives which may either follow the EDIT call or be in a file named by the EDIT parameter EDTINPUT. An SMS $OPTION card is not required, as one is automatically supplied by EDIT ($OPTION A,B,D,L=1). If the user chooses to override the default $OPTION card, the A, B, and D options must still be specified to insure that all libraries will be properly updated. The user is also cautioned that a deck which is deleted from the source library (SRCLIB) is not automatically deleted from the object library (OBJLIB). This situation seldom arises, and may be remedied by using the EDITKILL macro to delete the object code.

## EDIT PARAMETERS

The format of the EDIT call is as follows:

```
/ØEDITØPATH[,IEDIT={0|1|2|3},MXVR={1|2|.|.|.|64},CAT={YES|NO},OLDOBJ={YES|NO},
OLDSRC={YES|NO},DTYP={TAPE|PAD},RATFOR={YES|NO},FTNOPT=(*),
FTVERS=*,SPACE=<integer>,FTNTIME=<integer>,FOSYS={YES|NO|REL|KEEP},
LNKOPT=(*),EDITLIST=<name>,LNKINPUT=<name>,EDTINPUT=<name>]
```

The quantities between the square brackets are all optional; the default values are underlined. The meaning of each of the parameters is given below.

PATH — Existing pathname of program to be edited.

IEDIT — 0 - Perform source update.
1 - Same as IEDIT=0, plus compile updated routines.
2 - Same as IEDIT=1, plus update object

library.
3 - Same as IEDIT=2, plus build a new load module.

MXVR — Number of versions of files to be kept. This parameter is meaningful only for the first EDIT run having CAT=YES.

CAT — Command to catalog new files if the run is successful.

OLDOBJ — Indicates whether object code has previously been cataloged by EDIT. Should be set to NO until after the first time EDIT is used with IEDIT 2 and CAT=YES. Thereafter the default value of OLDOBJ=YES may be used.

OLDSRC — Similar to OLDOBJ except that it applies to the source file; OLDSRC should be set to NO until after the first time EDIT is run with CAT=YES (i.e., a "creation run").

DTYP — File storage device desired. This parameter may be different for successive EDIT runs, but is meaningful only when CAT=YES.

RATFOR — Instructs EDIT that this run updates decks written in RATFOR. Each RATFOR program must be preceded by a card with a sharp sign in column 1 to initiate RATFOR processing.

FTNOPT, FTVERS, LNKOPT — See FTN and LNK macros for FORTRAN and LINKAGE EDITOR options. System default options will be supplied if these parameters are omitted.

SPACE, FTNTIME — Identical to FTN SPACE and FTNTIME parameters. System defaults will be supplied when SPACE or FTNTIME (or both) are omitted.

EDITLIST — Used to name print file. If this parameter is omitted, the print file name will be EDIT.PRT.

LNKINPUT — Names an optional input file to the LINKAGE EDITOR. This file is required only if the user desires to supply routines to the LINKAGE EDITOR which exist on external libraries.

EDTINPUT — Access name of file containing the SMS

directives. If this parameter is omitted, it is assumed that SMS directives immediately follow the EDIT call card.

FOSYS — Controls printing of the EDIT output file:
YES - Print output file.
NO - Do not print output file; it will remain with the access name EDIT.PRT unless a different access name was specified by using the EDITLIST parameter.
REL - Release the output file.
KEEP - Print the output file, but do not release it.

## EXAMPLES

### ESTABLISHING A PROGRAM LIBRARY

```
/ JOB
/ USERMAC
/ LIMIT BAND=30
/ EDIT EXAMPLE/PATH/NAME,OLDOBJ=NO,OLDSRC=NO,IEDIT=2
$MODSET EXAMPLE,USER=JOEUSER
$DECK EXMAIN,LANG=FORTRAN,USER=JOEUSER,ACTION=ADD
      PROGRAM EXMAIN
C
C *** MAIN PROGRAM FOR EXAMPLES
C
      X = 3.
      Y = 2.
      Z = X+Y
      CALL EXSUB(X,Y,Z)
      STOP
      END
$DECK DUMMY,LANG=FORTRAN,USER=JOEUSER,ACTION=ADD
      SUBROUTINE DUMMY
C
C *** THIS IS A DUMMY SUBROUTINE
C *** IT WILL BE REMOVED LATER
C
      RETURN
      END
/ EOJ
```

This example establishes a source library at pathname EXAMPLE/PATH/NAME/SRCLIB, and an object library at pathname EXAMPLE/PATH/NAME/OBJLIB. The node EXAMPLE/PATH/NAME was

previously cataloged by the user. The USERMAC statement allows the user's job to access the macro library which contains EDIT. IEDIT is set to 2 to inhibit load module creation, since the main program calls a subroutine not yet in the library. The OLDOBJ and OLDSRC parameters are both set to NO to indicate that neither OBJECT nor SOURCE libraries have previously been cataloged for this particular program node.

ADDING A DECK TO AN EXISTING LIBRARY

```
/ JOB
/ USERMAC
/ LIMIT BAND=30
/ EDIT EXAMPLE/PATH/NAME,RATFOR=YES
$MODSET UPDATE1,USER=JOEUSER
$DECK EXSUB,LANG=RATFOR,USER=JOEUSER,ACTION=ADD
#         THIS CARD FLAGS RATFOR DECK
   SUBROUTINE EXSUB(X,Y,Z)
C
C  SUBROUTINE FOR EXAMPLES
C
   R = X*X+Y*Y
   IF (R .EQ. 0.) RETURN     # THE SHARP SIGN
   IF (R .GE. Z)                 # ALLOWS USERS TO
      {                             # COMMENT MORE FREELY
      Z = SIN(Z)
      X = 0.
      Y = 0.
      }
   ELSE Z = COS(Z)
   RETURN
   END
/ EOJ
```

This example adds a deck to the program library created in the first example. The RATFOR=YES option is used to inform EDIT that this run contains references to DECKS which are written in RATFOR. The card with the sharp sign in column 1 is a flag to the RATFOR processor that RATFOR code follows. FORTRAN and RATFOR decks may be freely interspersed, since the END card of each RATFOR program turns the RATFOR processor off. The default EDIT option (IEDIT=3) is used in this case to allow a load module to be built, since this run supplies the subroutine which was missing in the first example. The load module will be cataloged at pathname EXAMPLE/PATH/NAME/LOADLIB.

MAKING A CORRECTION TO AN EXISTING PROGRAM

```
/ JOB
/ USERMAC
/ LIMIT BAND=30
/ EDIT EXAMPLE/PATH/NAME
$MODSET UPDATE2,USER=JANEUSER
$DECK EXMAIN,USER=JANEUSER
$REPLACE EXMAIN 7
      Z=X*Y
$DECK DUMMY,ACTION=DEL
/ EOJ
```

This example shows how the user would replace line 7 of the main program in the first example with the desired line, Z=X*Y. The source code for the dummy routine is also deleted. Since the default EDIT option is used (IEDIT=3), all libraries, including the load module, would reflect the change. However, the object code for the deck DUMMY will remain on the object library, as mentioned earlier. The EDITKILL macro example shows how to remove the unneeded object code.

EXECUTING AN EXISTING PROGRAM

```
/ JOB
/ ASG SYS.LMOD,EXAMPLE/PATH/NAME/LOADLIB,USE=SHR
/ FXQT
/ EOJ
```

This example shows how to execute the program developed in the previous three examples. Since the load module was assigned with the access name SYS.LMOD, the FXQT macro may be used to accomplish a standard FORTRAN execution.

## EDIT UTILITY MACROS

Several macros have been written which perform support functions not directly performed by the EDIT macro. These macros enable users to conveniently perform tasks such as automatically inserting $DECK cards into files of FORTRAN code, punching decks, etc. In the following descriptions,

the quantities between the brackets are all optional; the default values are underlined.

EDITCATV

EDITCATV is the lower level macro which EDIT uses to catalog its files. Users may call EDITCATV explicitly after an EDIT execution to catalog the new libraries at a pathname different from the one used on the EDIT call. The CAT=NO option must be used to accomplish this. There are no defaults for any of the EDITCATV parameters.

EDITCATV PARAMETERS

/ØEDITCATVØPATH,IEDIT,DTYP,MXVR

| | |
|---|---|
| PATH | An existing pathname. The new EDIT files will be cataloged beneath this node using the standard EDIT names SRCLIB, OBJLIB, and LOADLIB. |
| IEDIT | Must be identical to the value of IEDIT used in the EDIT execution which preceeds EDITCATV. |
| DTYP | Device type for the new EDIT files. Legal values are TAPE or PAD. |
| MXVR | Integer which specifies the maximum number of files to be kept as backup. |

EDITCATV EXAMPLE

```
/ JØB
/ USERMAC
/ LIMIT BAND=3Ø
/ EDIT EXAMPLE/PATH/NAME,CAT=NØ
$MØDSET UPDATE3,USER=JANEUSER
$DECK EXMAIN,USER=JANEUSER
$REPLACE EXMAIN 5
      X = 4.
/ EDITCATV NEW/PATH/NAME,3,PAD,2
/ EØJ
```

This example modifies the old library and catalogs it at a new pathname. The old library will remain unchanged. The new files will reside on PAD, having a maximum of two versions. IEDIT=3 was selected to agree with the default value used in the preceeding EDIT execution.

## EDITDECK

EDITDECK is used to punch a deck of a program maintained by EDIT.

EDITDECK PARAMETERS

/ØEDITDECKØPATH [,FOSYS={YES / NO},DECKNAME=<name>,VERS=<integer>]

| | |
|---|---|
| PATH | Pathname of the program libraries. |
| FOSYS | YES - Produce a deck of punched cards.<br>NO - Produce a file called COMPILE |
| DECKNAME | Used to name a specific deck in the SRCLIB for which source code is desired. If DECKNAME is not specified, all decks on the SRCLIB will be selected. |
| VERS | Version of the SRCLIB to be used. |

EDITDECK EXAMPLE

```
/ JØB
/ USERMAC
/ LIMIT BAND=30
/ EDITDECK EXAMPLE/PATH/NAME
/ EØJ
```

The above example shows how to obtain a punched deck of the source library built in the previous examples.

## EDITFIX

EDITFIX is used when the current version of a program becomes unusable for some reason (faulty tape, etc.). It replaces the current version with an earlier version. If the VERS parameter is set to +0, EDITFIX will replace the current version with a new copy of the current version. This feature may be used to move tape resident files to disc, or to make a new copy of an often-used tape.

EDITFIX PARAMETERS

```
/ØEDITFIXØPATH[,IEDIT={0|1|2|3},VERS={integer|-1},DTYP={TAPE|PAD}]
```

| | |
|---|---|
| PATH | Pathname of the program libraries. |
| IEDIT | Specifies which of the program library files are to be recataloged. |
| VERS | Specifies the version number of the old version to be used. The default value of -1 "backs you up" one version. |
| DTYP | Device type of new file. |

EDITFIX EXAMPLE

```
/ JØB
/ USERMAC
/ LIMIT 8AND=3C
/ EDITFIX EXAMPLE/PATH/NAME
/ EØJ
```

The above example shows how an EDIT user could move his program libraries back one version if a file somehow became damaged.

EDITKILL

The EDITKILL macro is used to remove an unwanted member from the current OBJLIB. This is necessary if the user either changes the name of or deletes a function, subroutine, or main program on the source library.

EDITKILL PARAMETERS

/ØEDITKILLØPATH,MEM [,DTYP= {TAPE / PAD}]

| | |
|---|---|
| PATH | Pathname of the program libraries. |
| MEM | Name of the member to be deleted from the OBJLIB. |
| DTYP | Device type of the new OBJLIB. |

EDITKILL EXAMPLE

```
/ JOB
/ USERMAC
/ LIMIT BAND=30
/ EDITKILL EXAMPLE/PATH/NAME,DUMMY
/ EOJ
```

The above example deletes the routine called DUMMY from the OBJLIB built in the previous examples.

SMSCNVT

The SMSCNVT macro is used to insert a $DECK card before each routine in a sequential FORTRAN source file. It also permits users to specify files which will be inserted before or after the output file produced by SMSCNVT.

SMSCNVT PARAMETERS

/ØSMSCNVTØINFILE,OUTFILE [,FRONT=<name>,BACK=<name>,

ACTION={ADD / OTHER},LANG={FORTRAN / OTHER},USER={NONE / OTHER}]

| | |
|---|---|
| INFILE | Access name of FORTRAN source file. |
| OUTFILE | Access name of file to be produced which contains the $DECK cards before each routine. |

FRONT — Access name of file which will be copied onto the begining of OUTFILE. If data cards follow the SMSCNVT call card, these cards will be used as the FRONT data file.

BACK — Access name of file to be copied onto the end of the OUTFILE.

ACTION, LANG, USER — Parameters for the $DECK cards.

SMSCNVT EXAMPLE

```
/ JOB
/ USERMAC
/ START ACNM=CARDDECK
      PROGRAM EXAMPL
C
C ... FORTRAN STATEMENTS ...
C
      END
      SUBROUTINE A
C
C ... MORE FORTRAN ...
C
      END
      FUNCTION B(DUM)
C
C ... MORE FORTRAN ...
C
      END
/ STOP
/ SMSCNVT CARDDECK,EDITIN,USER=MORAWSKI
$MODSET TESTCASE
/ FOSYS EDITIN
/ EOJ
```

This example shows how to use the SMSCNVT macro. The output file produced by the example is shown on the next page.

OUTPUT FROM THE SMSCNVT EXAMPLE

```
$MODSET TESTCASE
$DECK A        ,LANG=FORTRAN ,ACTION=ADD ,USER=MORAWSKI
      SUBROUTINE A
C
C ... MORE FORTRAN ...
C
      END
$DECK B        ,LANG=FORTRAN ,ACTION=ADD ,USER=MORAWSKI
      FUNCTION B(DUM)
C
C ... MORE FORTRAN ...
C
      END
$DECK EXAMPL   ,LANG=FORTRAN ,ACTION=ADD ,USER=MORAWSKI
      PROGRAM EXAMPL
C
C ... FORTRAN STATEMENTS ...
C
      END
```

COMPOSITE EXAMPLE

The following example shows how to use SMSCNVT and EDIT together to produce program libraries directly from a FORTRAN deck.

```
/ JOB
/ USERMAC
/ LIMIT BAND=30
/ START ACNM=CARDDECK
      PROGRAM EXAMPL
C
C ... FORTRAN STATEMENTS ...
C
      END
      SUBROUTINE A
C
C ... MORE FORTRAN ...
C
      END
      FUNCTION B(DUM)
C
C ... MORE FORTRAN ...
C
      END
/ STOP
/ SMSCNVT CARDDECK,EDITIN,USER=MORAWSKI
$MODSET TESTCASE
/ EDIT PATHNAME,OLDOBJ=NO,OLDSRC=NO,EDTINPUT=EDITIN
/ EOJ
```